Bab: A novel algorithm for training clean model based on poisoned data

Chen Chen[1,*], Haibo Hong[1], Tao Xiang[2], Mande Xie[3] and Jun Shao[1]

[1]*School of Computer Science, Zhejiang Gongshang University (ZJSU), 310018*
[2]*School of Computer Science, Chongqing University (CQU), 400044*
[3]*School of Information and Electronic Engineering, Zhejiang Gongshang University (ZJSU), 310018*

Abstract
Nowadays, machine learning is performing very well in the field of computer vision and natural language processing. However, recent research results indicate that machine learning models are extremely vulnerable to various malicious attacks, among which backdoor attacks are favored by attackers because of their easy deployment and high success rate. In fact, the attacker only needs to put a small amount of malicious data in the training dataset, so that the model triggers abnormal behavior under certain circumstances. In this work, we propose a BAB (backdoor against backdoor) algorithm for training a clean model on poisoned data. The BAB algorithm mainly relies on two characteristics of the backdoors: 1) Multiple backdoors can coexist well in the model 2) When there are multiple backdoors in the same model, the strongest backdoor can make the weaker backdoor ineffective. Therefore, we implant a backdoor in the poisoned dataset, and rely on the output performance to refine a training dataset that contains almost no poisoned data, so as to train a clean model with high accuracy. In the experimental part, we test five current mainstream backdoor poisoning attacks. Our experimental results reveal that the BAB algorithm has a remarkable effect on filtering poisoned data: we succeed in obtaining a clean dataset containing less than 0.1% poisoned data, and train a high-precision model with this dataset. Our code is open source in https://gitee.com/dugu1076/bab-algorithm.git.

1. Introduction

At present, neural networks are gradually applied in various fields such as image classification[1, 2, 3] and natural language processing[4, 5]. Meanwhile, these ubiquitous deep learning systems indeed induce various security problems, such as evasion attacks[6, 7, 8], model stealing attacks[9, 10], membership inference attacks[11], and backdoor attacks[12, 13], etc. Malicious attackers can utilize these attacks to steal private information or even make the system misjudgment in some cases, resulting in immeasurable losses. In this article, we focus more on the backdoor poisoning attack. Compared with ordinary data poisoning attacks[14, 15, 16], the backdoor poisoning attack does not affect the accuracy of the original task, but adds backdoors to the model that are only triggered in specific situations. The backdoor poisoning conditions are very easy to implement: contaminate part of the training dataset (such as adding a patch) and modifying the contaminated data label, a simple backdoor attack is done [17]. The trained model behaves the same as the normal model when it encounters benign input, but when non-benign input (with triggers) is provided, the model behaves abnormally. To make matters worse, due to the deepening of the model depth, training a high-precision neural network model often requires a large dataset. Many trainers rely on crawlers or third-party purchases to obtain training data, which gives the attackers many opportunities to carry out the backdoor poisoning attack. Unfortunately, most of the existing defense methods are based on anomaly checking of the trained model and then repairing the anomalous model [18, 19, 20], or filtering the anomalous output of the trained model [21], which are not applicable to the stage when the model has not yet been trained. In order to reduce the losses caused by such attacks, we wonder: Is it possible to isolate a completely clean dataset from the poisoned dataset and employ it to train a clean model?

Intuitively, this is not a very simple task. One reason is the unexplainability of neural networks. The essence of the neural network model is the combination of linear transformation and nonlinear transformation between matrices, and these single transformations have no practical and specific meaning, which also makes it impossible to detect abnormalities directly from the internal parameters of the model. In addition, the constant update of the backdoor poisoning attack renders the means of manual verification ineffective. Early backdoor attacks[17] have obvious disadvantages in that the triggers can be separated by human eyes. However, with the deepening of research, SIG[22], Refool[23], CBA[24] and other attacks have been introduced. The triggers and target labels of such poisoned data are integrated into the dataset in a very reasonable way, which makes manual verification impossible.

In this paper, we propose a backdoor against backdoor (BAB) algorithm that is able to filter a clean dataset and train a clean model without any prior knowledge of the backdoor data distribution in the dataset. We divide the

SafeAI'23: The AAAI's Workshop on Artificial Intelligence Safety, Feb 13-14, 2023, Washington, D.C.
chenchen990404@163.com (C. Chen)

task of training a clean model into two stages. The first stage is the filtering of clean dataset. In this stage, we take advantage of two inherent characteristics of backdoor attacks to distinguish clean data from poisoned data. The second stage is the standard model training process using the filtered clean dataset. Our main contributions are as follows:

- We put forward a new perspective on the coexistence of multiple backdoors and exploit the inherent characteristics among multiple backdoors as a basis for filtering poisoned data: multiple backdoors can coexist well in the model; when there are two backdoor triggers in one input, the more aggressive backdoor can make the weaker one fail;
- We advance the BAB algorithm to enable training clean models from poisoned data. We discuss the algorithm in detail and display the parametric performance in the experimental section;
- We apply the BAB algorithm to two standard public datasets, CIFAR-10 and GTSRB, and test it against five mainstream backdoor data poisoning attacks (three dirty label attacks and two clean label attacks). The experimental results are exciting and we successfully obtain a clean dataset with a poisoning rate of less than 0.1%, and obtain a clean model with high accuracy;

2. Related Work

This section mainly introduces several backdoor attack methods and defense knowledge that rely on the backdoor poisoning attack.

2.1. Backdoor Poisoning Attack

The backdoor poisoning attack mainly relies on introducing some malicious data into the training dataset, which is consistent with the normal model training during the model training phase. Existing backdoor attacks are mainly divided into two categories: 1) dirty label attacks 2) clean label attacks. The earliest dirty label attacks [17, 25, 24] mainly rely on modifying the label and adding a trigger, such as a single pixel, a square or a more complex pattern, but these simple attack methods are often found by manual inspection. To increase the stealth of the backdoor attack, the attacker optimizes the trigger to incorporate it into the clean data in a reasonable form, such as invisible noise and mixed mode. Unlike dirty label attacks, clean label attacks aim to optimize labels to bypass manual verification of labels, that is, to achieve attack results without modifying labels. Such attacks can bypass most existing detection schemes due to their weak aggressiveness.

In order to verify the performance of our BAB algorithm, we select three representative dirty label attacks: BadNets[17], Blend[25] and CBA[24], and two representative clean label attacks: SIG[22] and Refool[23] in this paper.

2.2. Defense

The defense against backdoor attacks mainly consists of two aspects. 1) Training dataset[26] 2) Model neuron[19, 26, 18].

Training dataset Backdoor defense methods based on training datasets are mostly detection methods, not repairing methods. To the best of our knowledge, there is currently no method to almost completely separate the poisoned data from the poisoned dataset and use it to train a clean model. This is mainly due to 1) For the detectors, the amount of poisoned data, triggers and attack patterns are unknown. 2) The threshold for backdoor triggering is extremely low. Even if most of the poisoned data is filtered, the remaining poisoned data may still trigger the backdoor. These issues make defending against backdoors from training datasets difficult.

Model neuron Most of the existing backdoor defense methods are based on anomaly detection of model neurons and repair the anomaly model. But this kind of defense is not practical. On one hand, the backdoor task and the original task are not completely separated in neurons, which leads to the loss of the original task when removing the backdoor task. On the other hand, repairing requires the original dataset or a small amount of clean datasets, which is not realistic in some cases.

In this paper, we put forward the BAB (backdoor against backdoor) algorithm. Unlike most existing defense methods, our method does not detect and repair model neurons. Instead, we filter the poisoned data directly from the data source, and employ the filtered data to train a clean model. Our algorithm bridges the gap in the field of backdoor defense from the training dataset. Moreover, compared with the model neuron-based inpainting method, our BAB algorithm has less loss for the original task and only needs the original dataset.

2.3. NAD

NAD[27] is a proven and effective way to remove backdoors, and it mainly on a small number of clean dataset and uses model distillation to fine-tune the attention mechanism of the teacher model, so that the teacher model no longer pays attention to the backdoor area, so as to eliminate the backdoor.

3. Problem Statement

3.1. Threat Data

Considering that most of the existing data comes from crawlers or untrusted third-party, we can not control over the information of the data. Hence, in this paper, we set the most favorable conditions for the attackers, that is, the attackers completely control over the training dataset and can poison the dataset in any proportion and in any way. While as the defender, only this batch of data can be obtained, and the information such as the poisoning rate and poisoning method is unknown.

3.2. Assumption

In this section, we will take an example to elicit our hypothesis. Here, we take MNIST as the dataset and BadNets as the poisoning method. Suppose that the trainer has a poisoned dataset *D*, where *D* suffers two non-conflicting backdoor poisoning attacks ($D = D_{clean} \cup D_{poision\_1} \cup D_{poision\_2}$). The trainer draws a random proportion(such as 50%) of the dataset from *D* each time to train a model set $M = \{M_0, M_1, \ldots, M_N\}$. Selects d ($d \in D$) to input *M*, when it is a clean data sample ($d \in D_{clean}$), since training only uses a small amount of data, the output on model set *M* should be messy, as shown in Fig. 1(A). When there is only one backdoor trigger ($d \in D_{poision\_1} \cup D_{poision\_2}$), the output of the model set *M* should all point to the target activated by the trigger, as shown in Fig. 1(B). When there are two backdoor triggers ($d \in D_{poision\_1} \cap D_{poision\_2}$), the strength of the backdoor is not constant due to different training data, which also leads to the situation as shown in Fig. 1(C). The output of the model set *M* should be the target activated by the two triggers.

In view of the above facts, we speculate that if a certain proportion of known backdoors are put into a batch of poisoned data sets and randomly select data to train a batch of models, when backdoor triggers are added to the data, the models' judgment on clean data should all point to the newly added backdoor class, and for poisoned data, the output class should not only contain the newly added backdoor pointing target.

However, we must consider the following situations. If the backdoor generated by the poisoner is very weak, it may cause that even for the poisoned data, models all point to newly implanted backdoor classes. This results in the omission of poisoned data. Therefore, we need to control the strength of the implanted backdoor, insert a very weak backdoor into the model, but still can be successfully activated by the trigger. In this way, we can make the judgment between clean data and poisoned data.

4. Method

In this section, we will introduce the BAB algorithm in detail. Our algorithm is mainly divided into four steps: data preprocessing, the training of the verification models, reasoning and division of the dataset, and the training of the formal model. The specific algorithm is described as Algorithm 1.

Algorithm 1: BAB algorithm

```
Initialization: Target$\{y_0, y_1, \ldots, y_n\}$,
$M_{ver} \in \{M_{ver\_0}, M_{ver\_1}, \ldots, M_{ver\_N}\}$,Trigger $\gamma$,
$Data_1\{x_0, x_1, \ldots\} \longmapsto$ Original Data
$Data_2\{x_0, x_1, \ldots\} \longmapsto$ Partial $Data_1$ Carrying
  Triggers $\gamma$ to attack target $n+1$
$Data_3\{x_0, x_1, \ldots\} \longmapsto$ All $Data_1$ Carrying
  Triggers $\gamma$;
for $M$ in $M_{ver}$ do
    for 1...epochs do
        $M$.forward($D_2$);
        loss=$\mathcal{L}_{BAB}$;
        loss.backward();
    end
end
for $x$ in $Data_3$ do
    for $M$ in $M_{ver}$ do
        if $M(x)! = n+1$ then
            Poisioned Data $\longmapsto$ remove
        else
            Continue;
        end
    end
end
Return Clean Dataset
```

4.1. Data Preprocessing

Firstly, we need to preprocess the data, as shown in Fig. 2. We extract a small portion (such as 10%) of the data, add triggers of arbitrary shape, and modify the model labels to new classes (preventing the same targets as poisoning attacks) and shuffle the data to generate a new dataset. After that, we randomly select N small parts (such as 50%) of the dataset as training data. Besides, we need an entire dataset plus this trigger for inference and partitioning the data in reasoning and division of the dataset.

4.2. Verification Model Training

Secondly, we need to train a batch of verification models, as shown in Fig 3. Put the dataset generated in the previous step into a batch of simple network models for a small number of iterations. In order to create a backdoor that is as weak as possible but can be successfully

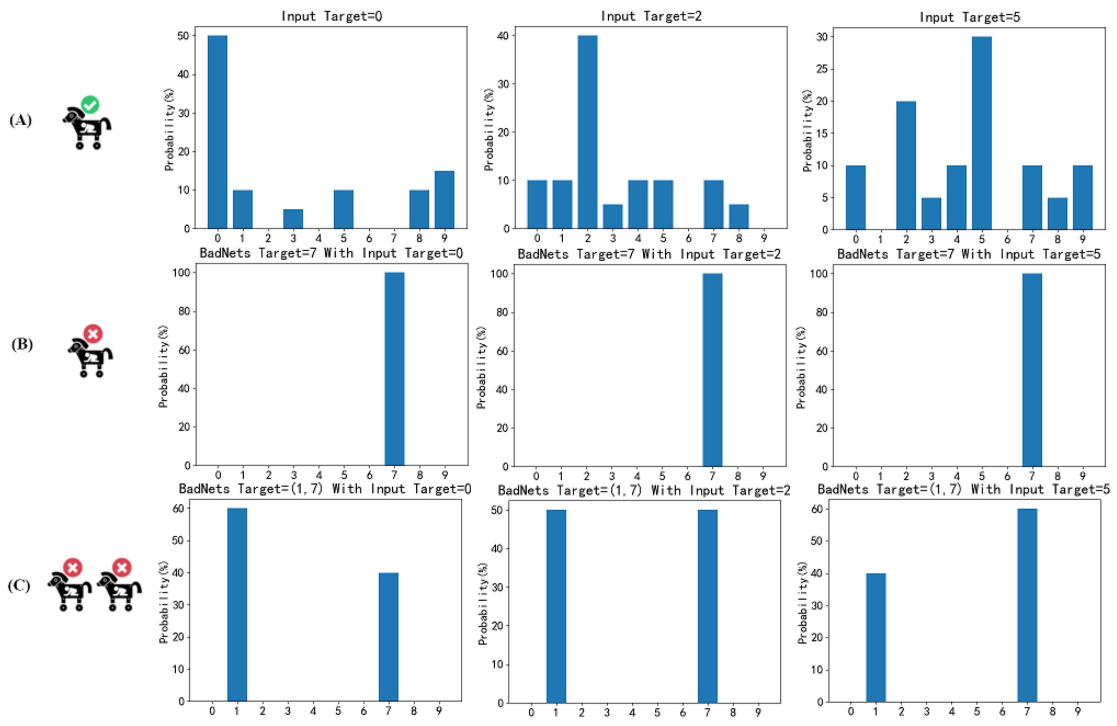

Figure 1: The performance of different categories of data in a batch of models.

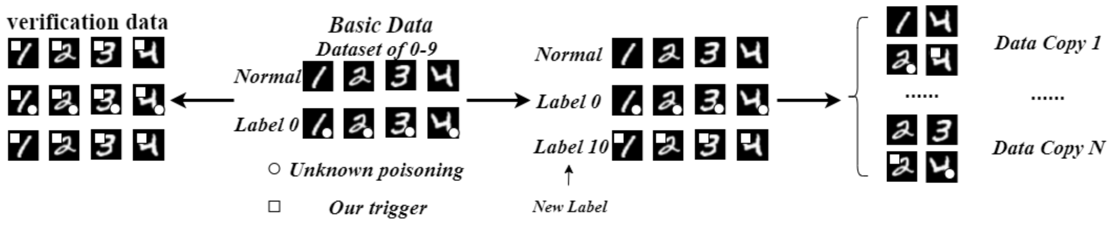

Figure 2: Data preprocessing.

activated, we reduce the neuron activation degree gap as much as possible when inputting poisoned data and clean data. In addition, We choose one or more layers of neurons to suppress their activation, and make certain improvements on the original loss function, just like Equation 1.

$$\mathcal{L}_{BAB} = \begin{cases} l(f(x), y) & \text{Clean} \\ l(f(x_{tri}), y_{tri}) + \alpha * L_2(\theta, \theta_{tri}) & \text{Backdoor} \end{cases} \tag{1}$$

where f is the trained model; y and y_{tri} are the original target and the backdoor attack target, respectively; θ and θ_{tri} are the activation values of clean data and backdoor data, respectively; α is a hyper-parameter used to coordinate the activation of inhibitory neurons. In Equation 1, $l(f(x_{tri}), y_{tri})$ ensures that the backdoor can be triggered correctly, and $L_2(\theta, \theta_{tri})$ minimizes the gap between the backdoor data and the clean data in the neural network, so that the backdoor we generate is as weak as possible. After extensive experiments, we find that fitting the penultimate layer (the previous layer of the softmax) works best in the same network layer.

4.3. Inference

The most important step is the division of poisoned data and clean data, as shown in Fig. 4. Through simple training, we get a batch of simple neural network verification models. Then, we feed the verification data sequentially into the verification model. When the verification data

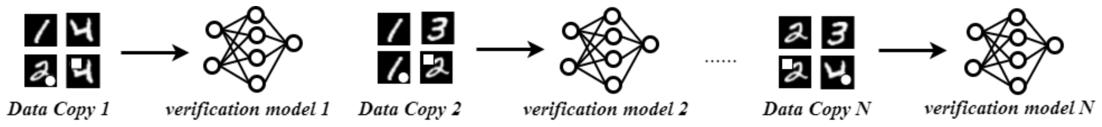

Figure 3: Simple training of the verification model.

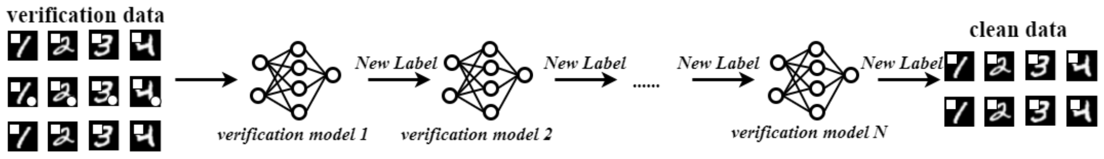

Figure 4: The division of poisoned data and clean data.

passes all verification models, we consider the data to be clean; otherwise, the data is illegally poisoned by others. There will be some accidental injuries due to the accuracy of the implanted backdoor, but this is inevitable. In subsequent experiments, we find that these accidental injuries are acceptable in small amounts.

4.4. Formal Model Training

After the above steps, a batch of clean data can be obtained, and a clean model can be obtained by training in a standard way using this clean dataset.

5. Experiment

5.1. Experimental Setup

All experiments are run on a hardware equipped with a RTX 3070 GPU and an i7 10700K CPU.

Attack Configurations We consider 5 backdoor attacks in our experiments, including three dirty label attacks: BadNets[17], Blend attack[25] and composite backdoor attack (CBA)[24], two clean label attacks: natural reflection (Refool)[23]. and sinusoidal signal attack (SIG)[22]. We follow the settings suggested by these papers and the open-sourced code corresponding to their original papers to configure these attack algorithms. All attacks are evaluated on two benchmark datasets, CIFAR-10[28] and GTSRB[29], with a classical model structures including ResNet-18[2]. For the backdoor poisoned data, we train the backdoor model for 100 epochs using the Adam optimizer and the learning rate is set to be 0.01. Considering the uneven distribution of the GTSRB dataset, we set the target label of the SIG and Refool poisoning attacks to be 1, and the target label of the rest of the poisoning attacks to be 0. SIG[1] and Refool[2] both adopt the open source code of the original paper. We have not exploited any data augmentation techniques to avoid side effects on attack success rate. In subsequent experiments, we mainly take the CIFAR-10 dataset as the test dataset because its data distribution is more uniform.

[1]https://github.com/bboylyg/NAD

[2]https://github.com/DreamtaleCore/Refool

Defense and Training Details We compare our BAB with a state-of-the-art defense method: Neural Attention Distillation (NAD)[27]. For NAD, we follow the configuration specified in original papers.

NAD We take open source code [3] as a base for extensions. We try to keep the parameters consistent with our experiments, including model architecture, learning rate, number of iterations, etc. In addition, following the recommendations of [27], we set the proportion of clean data owned by NAD to be 5%, the number of iterations when acquiring the teacher model to be 10. When using the teacher model to clean the student model, we set the number of iterations to be 100, low layer β=500, middle layer β=1000, high layer β=1000.

BAB On the CIFAR-10 dataset, we set N=10, α=0.2, R=0.5, and use the Adam optimizer to train the verification models for 5 epochs, set the learning rate to be 0.01, the number of iterations of each verification model to be 5, and the target of the model embedded in the verification model to be 10. On the GTSRB dataset, we set N=5, α=0.3, R=0.5, and use the Adam optimizer to train the verification models for 5 epochs, setting the learning rate to be 0.01, the number of iterations for each verification model to be 5, and the model embedded in the verification models to set a target of 43. In the training phase of the formal model, we set the model with a learning rate of 0.001 and an iteration number of 100 epochs. We have not used any data augmentation techniques to avoid side effects on attack success rate.

Evaluation Metrics We employ two commonly used performance metrics: Attack Success Rate (ASR), which

[3]https://github.com/bboylyg/NAD

is the classification accuracy on the backdoor test set, and Clean Accuracy (CA), which is the classification accuracy on the clean test set. In addition, we calculated the residual retention rate of clean data (CDR) and the residual rate of poisoned data(PDR).

5.2. Comparison to Existing Defenses

Table 1 and Table 2 reveal the experimental results of our proposed BAB on CIFAR-10 and GTSRB. We consider 5 state-of-the-art backdoor attacks and compare BAB to the current best-defense NAD. As defenders, we have two goals: maintain clean accuracy and reduce attack success rate; high attack success rate will lead to the possibility of illegal usage of the model, and low accuracy will make the model useless.

As described in Table 1, we can see that on CA we are on par with the NAD method at CIFAR-10 (77.81% vs 77.60%) and GTSRB (91.22% vs 90.81%). But for ASR, our elimination rate for backdoors is much better than NAD, especially on GTSRB dataset (2.54% vs 21.76%). Compare the CA of the model generated by the BAB algorithm with the CA of the model generated by the clean data (as shown in row None and column "No Defense"), we find about 5% performance loss on CIFAR-10 dataset (77.81% vs 83.34%) and about 2% performance loss on GTSRB dataset (91.22% vs 92.92%). We believe that this part of the performance loss is mainly due to the reduction of training dataset, part of it comes from the attacker's poisoning, and part of it comes from the accidental injury of the BAB algorithm. In Table 1, we can see that we lose about 10% of the clean dataset, and almost completely remove the poisoned data. This part of the loss, as a limitation of the BAB algorithm, will continue to be studied in future work.

5.3. Number of Verification Models

Here, we investigate the effect of the number N of verification models on filtered clean datasets versus residual poisoned data on CIFAR-10. Our goal is to keep the clean dataset as much as possible while filtering the poisoned data, so that a clean and more accurate model can be trained in the formal training phase. We run the BAB algorithm on N belonging to [1, 20] and display the amount of clean data and the amount of residual poisoned data in Fig. 5. Obviously, it is found that there is a trade-off between amount of clean data and amount of residual poisoned data. Specifically, as the number of models N increases, the clean dataset will be lost along with the poisoned dataset, we find that the loss of clean data sets is mainly due to the suppression of neurons in the implanted backdoor, since not every implanted backdoor can reach a 100% attack success rate, this causes some data to be mistaken for poisoned data and discarded. In addition, we find that when N>10, the poisoned data has almost been filtered out, the clean data of Refool and SIG attacks are gradually lost, while the data is relatively stable in the other three attacks. We believe that it is because Refool and SIG are clean label attacks that only mix the trigger pattern (i.e. superimposed sinusoidal signal or natural reflection) with the background of the poisoned image, which makes this type of attack relatively weak, resulting in mass misjudgments. In fact, the BAB algorithm with the number of models N=10 is sufficient to withstand these five attacks, even when the backdoor poisoning rate is extremely high, i.e. 70%, or a variety of backdoor attacks (see Section. 3).

5.4. Hyper-parameter α

Here, we investigate the effect of hyper-parameter α on clean data and residual poisoned data, we compare the different effects of five values on the CIFAR-10 dataset: 0, 0.001, 0.002, 0.0003 and 0.005, the experimental results are displayed in Fig. 6. We find that repression of neurons is necessary to weaken the ability of our implanted backdoors. Compared with the filtering efficiency of the BAB algorithm without suppression (None) and with suppression for poisoning data, the efficiency of suppression is much higher than that of the BAB algorithm without suppression. In addition, it is not appropriate to suppress the implantation of backdoors too much. Although this can significantly improve the efficiency of poisoning data, it can lead to a large amount of clean data loss due to low attack success rate (ASR). Fortunately, since the implanted backdoor is controlled by the defender, the attack success rate (ASR) is visible, so it is desirable to suppress neuronal activation while maintaining a high attack success rate (ASR).

5.5. Pressure Test

Here, we test when BAB encounters some extremes. Now we know that the BAB algorithm can filter the poisoned data well and train a clean model.

Therefore, the challenge for the BAB algorithm is whether the BAB algorithm can still filter out a clean dataset at a small cost and train a clean model when it encounters a large proportion of poisoning or there are multiple poisoning attacks. We experiment on 3 attacks, BadNets, Blend and CBA on CIFAR-10, with poisoning rates up to 50%/70%, and show the results in Table 3. In addition, we also test for mixed attacks, and the total poisoning rate is as high as 50%/70%, and the results are shown in Table 4. We find that even at 70% poisoning rate, our BAB algorithm successfully reduces attack success rate (ASR) from 99.67% to 3.23% for BadNets, 93.18% to 4.89% for CBA, and 100% to 5.15% for Refool, respectively. For the mixed attacks, BAB also successfully reduces the

Table 1
The attack success rate (ASR%) and the clean accuracy (CA%) of 2 backdoor defense methods against 5 backdoor attacks including 3 dirty label attacks and 2 clean label attacks. None means the training data is completely clean. The best results are in bold.

Dataset	Types	No Defense		NAD		BAB(ours)	
		ASR	CA	ASR	CA	ASR	CA
CIFAR-10	None	0%	83.34%	-	-	-	-
	BadNets	97.42%	82.67%	**0.30%**	78.67%	1.88%	**79.17%**
	Blend	100%	65.81%	4.50%	**78.02%**	**0.12%**	77.20%
	CBA	79.01%	83.26%	7.06%	**77.96%**	**2.08%**	77.70%
	SIG	99.99%	82.12%	**0.01%**	76.14%	**0.01%**	**77.37%**
	Refool	99.93%	82.68%	0.07%	77.21%	**0%**	**77.61%**
	Average	95.27%	79.31%	2.39%	77.60%	**0.82%**	**77.81%**
GTSRB	None	0%	92.92%	-	-	-	-
	BadNets	93.07%	88.25%	3.23%	**93.68%**	**0.90%**	92.24%
	Blend	96.32%	90.58%	14.68%	**89.74%**	**7.14%**	89.31%
	CBA	89.93%	90.93%	29.95%	91.08%	**4.68%**	**91.80%**
	SIG	100%	91.24%	12.33%	87.72%	**0%**	**90.92%**
	Refool	91.64%	85.90%	19.95%	89.76%	**0%**	**91.82%**
	Average	94.19%	89.97%	21.76%	90.81%	**2.54%**	**91.22%**

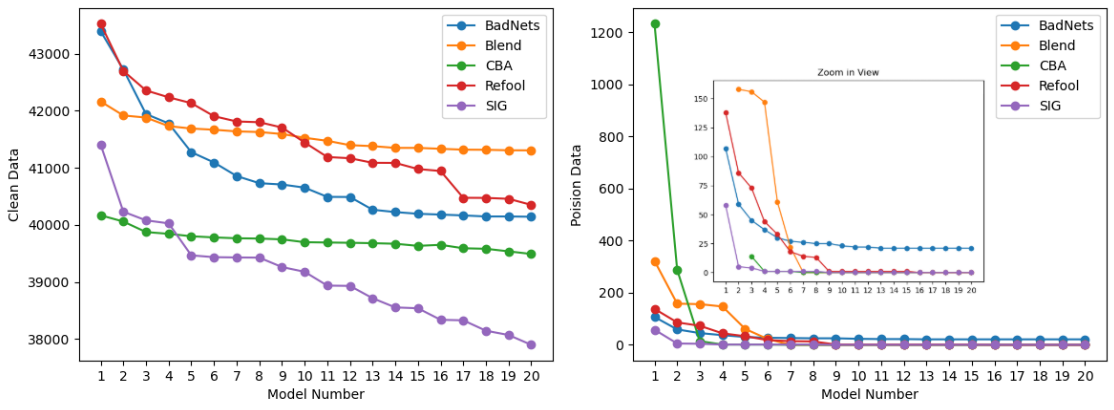

Figure 5: Performance of our BAB with different verification model number N ∈ [1, 20] on CIFAR-10 dataset. Left: Number of clean data (CD); Right: Number of residual poisoned data (PD).

attack success rate (ASR) from more than 90% to less than 5% with the mixed attack of BadNets and Refool and the mixed attack of BadNets and CBA. In addition, we find that in the mixed attacks, the accuracy of the original task increases after applying the BAB algorithm. We believe that this is due to the existence of multiple poisoning attacks in the dataset, which limits the training accuracy of the model. After the BAB algorithm filters out poisoned data, this limitation is broken. Overall, our BAB algorithm has good robustness.

6. Conclusion

In this paper, we propose a novel algorithm to train clean models on poisoned data. Firstly, we implant our own backdoor in the detected dataset, and train multiple verification models, relying on comparing the outputs of the verification models to divide the clean data from the poisoned data. Secondly, we train a formal model with the partitioned clean dataset. We apply our algorithm to two different datasets, experimenting with five attack

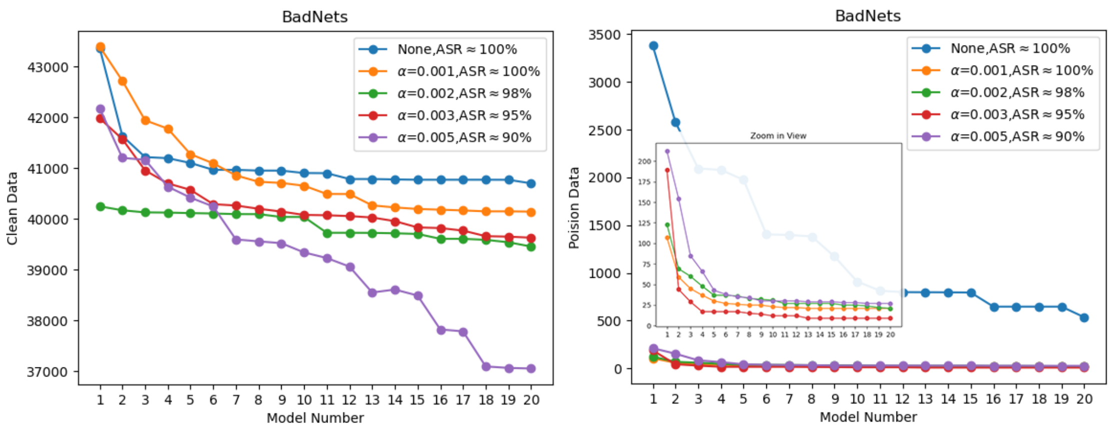

Figure 6: Performance of our BAB with different Hyper-parameter α on CIFAR-10 dataset. Left: Number of clean data (CD); Right: Number of residual poisioned data (PD).

Table 2
The residual retention rate of clean data (CDR) and the residual rate of poisoned data (PDR) after BAB against 5 backdoor attacks including 3 dirty label attacks and 2 clean label attacks.

Dataset		BadNets	Blend	CBA	SIG	Refool
CIFAR-10	CDR	91.70%	87.85%	89.33%	90.59%	90.11%
	PDR	0.28%	0%	0%	0%	0.02%
GTSRB	CDR	82.75%	82.74%	85.74%	90.43%	88.67%
	PDR	1.40%	1.32%	0%	0%	0%

modalities. The experimental results indicate that our algorithm is useful and effective. Subsequently, we also analyze and discuss how to choose the parameters reasonably and the robustness of the algorithm. Overall, our work provides a feasible direction for training clean models on poisoned data.

Acknowledgments

This work is partially supported by the National Natural Science Foundation of China (NSFC) (Grant Nos.61602408,61972352,U1709217) and Zhejiang Provincial Natural Science Foundation of China under Grant (Nos.LY19F020005, LY18F020009).

References

[1] Q. Dong, X. Zhu, S. Gong, Single-label multi-class image classification by deep logistic regression, in: Proceedings of the AAAI conference on artificial intelligence, volume 33, 2019, pp. 3486–3493.

[2] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[3] L. Brigato, B. Barz, L. Iocchi, J. Denzler, Image classification with small datasets: Overview and benchmark, IEEE Access (2022).

[4] M. Mridha, A. Q. Ohi, M. A. Hamid, M. M. Monowar, A study on the challenges and opportunities of speech recognition for bengali language, Artificial Intelligence Review 55 (2022) 3431–3455.

[5] A. Romanenko, Robust speech recognition for low-resource languages, Ph.D. thesis, Universität Ulm, 2022.

[6] W. Wang, B. Yin, T. Yao, L. Zhang, Y. Fu, S. Ding, J. Li, F. Huang, X. Xue, Delving into data: Effectively substitute training for black-box attack, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 4761–4770.

[7] Y. Yu, X. Gao, C.-Z. Xu, Lafeat: Piercing through adversarial defenses with latent features, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 5735–5745.

[8] C. Ma, L. Chen, J.-H. Yong, Simulating unknown target models for query-efficient black-box attacks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 11835–11844.

[9] S. Kariyappa, A. Prakash, M. K. Qureshi, Maze: Data-free model stealing attack using zeroth-order gradient estimation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 13814–13823.

[10] L. Lyu, X. He, F. Wu, L. Sun, Killing two birds with one stone: Stealing model and inferring attribute from bert-based apis, arXiv preprint arXiv:2105.10909 (2021).

[11] A. Salem, Y. Zhang, M. Humbert, M. Fritz, M. Backes, Ml-leaks: Model and data independent

Table 3
Pressure testing with poisoning rate up to 50% and 70% for 3 attacks including BadNets, Blend, and CBA on CIFAR-10 dataset.

Poisoning Rate	Defense	BadNets		Blend		CBA	
		ASR	CA	ASR	CA	ASR	CA
50%	None	99.41%	70.64%	99.90%	49.74%	88.57%	72.14%
	BAB	3.70%	69.36%	0.51%	71.50%	4.25%	69.97%
70%	None	99.67%	63.32%	100%	50.16%	93.18%	55.59%
	BAB	3.23%	62.58%	2.69%	65.43%	4.89%	63.03%

Table 4
Numerical values of clean and poisoned data after pressure tests including BadNets Hybrid CBA and BadNets Hybrid Blend with poisoning rates up to 50% and 70%.

Poisoning Rate	Defense	BadNets+Blend		BadNets+CBA	
		ASR	CA	ASR	CA
50%	None	98.21%+99.83%	61.86%	98.96%+99.49%	69.19%
	BAB	2.76%+0.12%	70.03%	2.53%+4.11%	70.26%
70%	None	98.78%+100%	59.29%	98.60%+99.41%	61.35%
	BAB	5.05%+0.81%	66.19%	2.44%+4.82%	65.61%

membership inference attacks and defenses on machine learning models, in: Network and Distributed Systems Security Symposium 2019, Internet Society, 2019.

[12] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, X. Zhang, Trojaning attack on neural networks (2017).

[13] Z. Xi, R. Pang, S. Ji, T. Wang, Graph backdoor, in: 30th USENIX Security Symposium (USENIX Security 21), 2021, pp. 1523–1540.

[14] N. G. Marchant, B. I. Rubinstein, S. Alfeld, Hard to forget: Poisoning attacks on certified machine unlearning, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 36, 2022, pp. 7691–7700.

[15] M.-H. Van, W. Du, X. Wu, A. Lu, Poisoning attacks on fair machine learning, in: International Conference on Database Systems for Advanced Applications, Springer, 2022, pp. 370–386.

[16] B. Zhao, Y. Lao, Clpa: Clean-label poisoning availability attacks using generative adversarial nets (2022).

[17] T. Gu, B. Dolan-Gavitt, S. Garg, Badnets: Identifying vulnerabilities in the machine learning model supply chain, arXiv preprint arXiv:1708.06733 (2017).

[18] K. Liu, B. Dolan-Gavitt, S. Garg, Fine-pruning: Defending against backdooring attacks on deep neural networks, in: International Symposium on Research in Attacks, Intrusions, and Defenses, Springer, 2018, pp. 273–294.

[19] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, B. Y. Zhao, Neural cleanse: Identifying and mitigating backdoor attacks in neural networks, in: 2019 IEEE Symposium on Security and Privacy (SP), IEEE, 2019, pp. 707–723.

[20] LiYige, X. Lyu, N. Koren, L. Lyu, B. Li, Anti-backdoor learning: Training clean models on poisoned data, Advances in Neural Information Processing Systems 34 (2021) 14900–14912.

[21] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, S. Nepal, Strip: A defence against trojan attacks on deep neural networks, in: Proceedings of the 35th Annual Computer Security Applications Conference, 2019, pp. 113–125.

[22] M. Barni, K. Kallas, B. Tondi, A new backdoor attack in cnns by training set corruption without label poisoning, in: 2019 IEEE International Conference on Image Processing (ICIP), IEEE, 2019, pp. 101–105.

[23] Y. Liu, X. Ma, J. Bailey, F. Lu, Reflection backdoor: A natural backdoor attack on deep neural networks, in: European Conference on Computer Vision, Springer, 2020, pp. 182–199.

[24] J. Lin, L. Xu, Y. Liu, X. Zhang, Composite backdoor attack for deep neural network by mixing existing benign features, in: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, 2020, pp. 113–131.

[25] X. Chen, C. Liu, B. Li, K. Lu, D. Song, Targeted backdoor attacks on deep learning systems using data poisoning, arXiv preprint arXiv:1712.05526 (2017).

[26] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, B. Srivastava, Detecting backdoor attacks on deep neural networks by activation clustering, in: SafeAI@ AAAI, 2019.

[27] Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li, X. Ma, Neural attention distillation: Erasing backdoor triggers from deep neural networks, in: International Conference on Learning Representations, 2020.

[28] A. Krizhevsky, G. Hinton, et al., Learning multiple layers of features from tiny images (2009).

[29] J. Stallkamp, M. Schlipsing, J. Salmen, C. Igel, Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition, Neural networks 32 (2012) 323–332.